Date of acceptance Grade

Instructor

Random Cookie protocol, a new solution to prevent against session cookie hijacking

Qijia Zeng

Helsinki May 4, 2016

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Qijia Zeng

Työn nimi — Arbetets titel — Title

Random Cookie protocol, a new solution to prevent against session cookie hijacking

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	May 4, 2016	56 pages + 0 appendices

Tiivistelmä — Referat — Abstract

With the rapid development of information technology, the network is playing a more and more important role in our life and penetrate to every corner of our life, such as responsible for computing and transacting in business industries, and make great contribution for social communication. However, at the same time, more and more network attacks appears and seriously endanger the security of network. Including the threads for wired network and wireless network. In this thesis, we first talk about network security, including wired network security and wireless security, we discuss the current threads and attacks we face for the wired network and wireless network, then we introduced the current solution for these threads and attacks.At last,we demonstrate the whole process of session cookie hijacking by manually perform it.We proposed our solution which called random cookie protocol, and we did experiments to compare the performance with HTTP.

ACM Computing Classification System (CCS):
A.1 [Introductory and Survey],
I.7.m [Document and text processing]

Avainsanat — Nyckelord — Keywords
layout, summary, list of references

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

With the rapid development of information technology, the network is playing a more and more important role in our life and penetrate to every corner of our life, such as responsible for computing and transacting in business industries, and make great contribution for social communication. However, at the same time, more and more network attacks appears and seriously endanger the security of network. Including the threads for wired network and wireless network. Now the use of cookie for storing short and important data is reaching to a universal level, such as session id, by keeping the session id for the session in web browser, can help web server to keep the status for this session with web browser, thus can avoid users to input username and password for every access to web server. However, the traditional HTTP protocol does not provide any security service, and the session cookie is transmitted over network in a plain way, this may cause the risk of being stolen by network adversaries, and we call this attack as session cookie hijacking. Now, some solutions to prevent against session cookie hijacking has been proposed like HTTPS, sessonlock protocol, however, although these solutions can provide some security for session cookie, they have their shortcoming on other aspects, like for HTTPS, it provide authentication, integration and confidentiality services, however, as it require expensive computing, lots of website just deploy it for the login page, and leave other pages under HTTP, and thus can give network adversaries chances to lunch network attacks.

In order to prevent against session cookie hijacking, in this thesis, we proposed a protocol called random cookie protocol, this method by using hash algorithm to generate a unique session cookie for every communication, once the session cookie is verified by web server, a new session cookie will be generated for next time use, so even if the session cookie is stolen by network adversaries, we don't need to worry about that because the session cookie is expired when it is verified by web server and can not be reused any more. Based on the random cookie protocol, we conducted some experiments to analyze its performance compared with normal HTTP protocol, the result revealed that although the performance of random cookie protocol is less than HTTP protocol, however, the difference of performance is slightly and can be acceptable, and we think the bonus of security guarantee is worth enough to offset the slight less performance.

So the rest of the thesis is like the following: in section 2, we discussed the network security, including the wired network security and wireless network security. In

section 3, we discussed web session, and some related concepts include what is cookie and session, the principle of cookie and session, and the session cookie hijacking, next we introduced the current solution for session cookie hijacking in section 4, such as HTTPS, sessionlock, one-time cookie and VPN, and also talked about their pros and cons. Follow by section 5, we give the demonstration of session cookie hijacking, and we proposed a new solution called random cookie project for session cookie hijacking in section6. Finally in section 6, we conclude the whole thesis.

2 Network security

Network plays a very critical role in our world. With the rapid development of information technology, network technology has become more and more sophisticated and has become an essential part of our modern life in this information era. It has penetrated into every corners of our life, and take responsibility for many important areas like conducting computations and transactions in business industries, or communications in government operations. However, threads always have afterward nature, when network technology is increasingly matured and popular among people's life, some networks attacks has appeared and some network security incidents has occurred and caused huge loss to companies and consumers. For example, in 2014 a security vulnerability called OpenSSL has been detected. The cause of OpenSSL is due to the failure of checking the boundary value of user's input before using the input as parameters when call memcpy() function. This security vulnerability is severe and cause the leakage of network user accounts and other privacy of network users. An other serious network security incident is the leakage of eBay data. In May 22th 2014, eBay suddenly required almost 128 millions eBay active users to reset their passwords for their eBay accounts as hackers can obtain passwords, telephone numbers, home addresses or other privacy of eBay users from eBay website. Although later eBay claimed the database which has been hacked and the information leaked didn't contain financial data such as credit card number, and won't cause huge loss for users, however, the increasingly network security incidents has began to gain people's attentions, and network security is becoming a frequent research topic. Network security refer to some operations or measurements that designed to protect network, more specifically, that is by some activities based on certain policy to ensure the reliability, confidentiality, integrity and safety of network, and prevent or stop a variety of network threads from entering or spreading the network. With the shift of type of network from wired network to wireless net-

work, the type of network thread is changed and new network thread appears by the time pass, and the network security protection mechanism is also changed. In the below, we will talk about the hardwired network security and wireless network security, among these, we will focus on the wireless LAN and discuss the type of WLAN attack.

2.1 Wired network

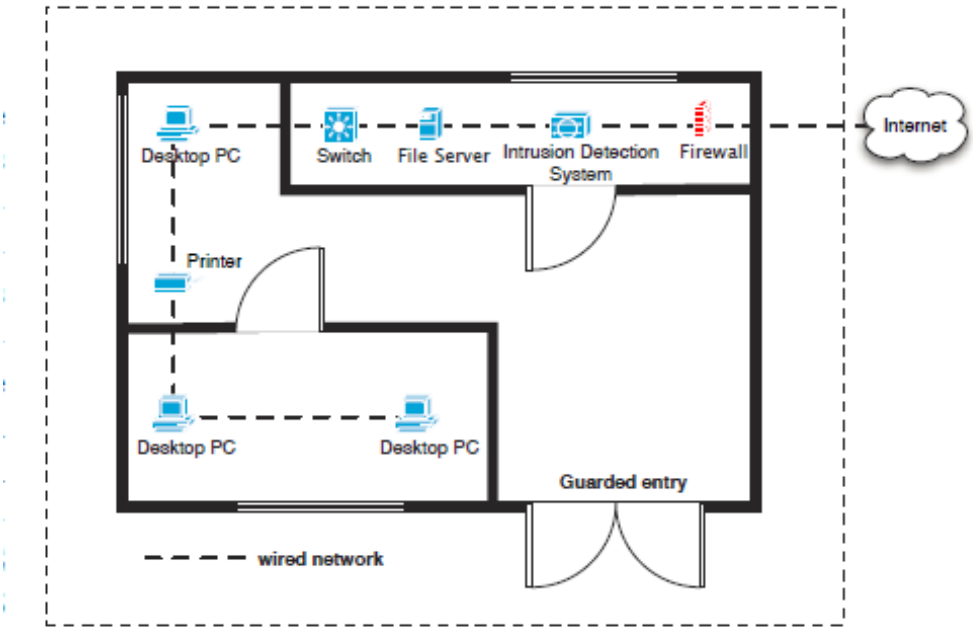

Figure 1: Wired Network

By the shape of network, we can categorize network as wired network and wireless network. Decades ago before the popular of wireless network, the traditional network is wired network. Different from wireless network, the wired network has its own physical shape, and each network components such as hosts or switches are connected by cables. From the Figure 1[Gun10] we can see that a wired network has its own physical scope, the desktop PC, printers and switches are connected by cables. at the entry of wired network with the Internet, a firewall is deployed

here to protect the wired network, so wired network is usually protected by firewalls. some state-of-the-art systems such as IDS(Intrusion detection system) and IPS(Intrusion prevention system) are designed as firewall systems to protect wired network. Compared with wireless network, wired network has its advantages and disadvantages. One point that may serve users better than wireless network is wired network can transfer information faster than wireless network,and the cost is relative cheap and affordable. However, with the increase of complexity of wired network such as adding more hosts into the wired network, the more expensive the network will cost. Another disadvantage is wired network take more physical spaces. As each component of wired network is connected by cables, the more complexity the wired network be, the more cables it will has, and thus will take spaces and limit your mobility.

2.2 Wireless network

with the development of information technology, the wireless network is becoming increasingly popular and companies is beginning to replace their wired network with wireless network. Different from wired network, wireless network has not a specific physical shape, and instead of transferring information by cable, wireless network transfer its information over the air. As air is uncontrolled and unprotected, and is shared with other wireless networks, it lacks the corresponding physical security measurements for the wired parts, so once a user connects a wireless access point(AP) to a network, the data can traverse through the walls or doors into to internal network directly, and may expose the structure of internal network to outside hackers, thus may cause serious threats to the network, and these traditional protection measurements for wired network such as IPS and IDS can not meet the security requirement any more. Usually, the general approach to ensure the security of wireless network is by adopting cryptograhic mechanisms to guarantee the security. Now for wireless network, there are three main cryptograhic standers that are IEEE 802.15.1(Bluetooth), IEEE 802.11(WIFI) and IEEE 802.16(WiMAX), there corresponds to personal-are, local-are and wide-area wireless networks.

2.3 Type of network attacks

The security technique of network is increasing along the development of technologies, however, the network risk is also enhanced and more and more network attacks

appeared to threat network. Below we will discuss some type of network attacks.

2.3.1 Eavesdropping

Eavesdropping a network layer attack consisting of capturing packets from the network transmitted by others' computers and reading the data content in search of sensitive information like passwords, session tokens, or any kind of confidential information. The eavesdropping attack is a serious security threat to a wireless network due to data in air since the eavesdropping attack is a prerequisite for other attacks. When an attacker is eavesdropping on your communications, it is referred to as sniffing or snooping. The ability of an eavesdropper to monitor the network is generally the biggest security problem that administrators face in an enterprise. Without strong encryption services that are based on cryptography, your data can be read by others as it traverses the network.

2.3.2 Data Modification

Once an attacker has read your data by eavesdropping, the next logical step is to alter them. An attacker can modify the data in the packet without the knowledge of the sender or receiver. Even if you do not require confidentiality for all communications, you do not want any of your messages to be modified in transit which could cause inestimable lose.

2.3.3 Identity Spoofing

IP address spoofing is one of the most frequently used spoofing attack methods. In an typical IP address spoofing attack, an attacker sends IP packets from a spoofed source address in order to disguise itself. Denial-of-service attacks(DoS attack) often use IP spoofing to overload networks and devices with packets that appear to be from legitimate source IP addresses.

Internet protocol is a network protocol operating at layer 3 of the OSI model. It is a connectionless which contains no information regarding transaction state, It is used to route packets on a network. Additionally, there is no method in place to ensure that a packet is properly delivered to the destination. By closely looking the IP header , we can see that the top 3 rows of the header which the first 96 bits contain various information about the packet. The next 64 bits the next 2 rows,

however, contains the source and destination IP addresses. Using one of several tools, an attacker can easily modify these addresses both source address and destination address specifically source field. IP spoofing is vulnerable because each datagram is sent independent of all others due to the stateless nature of IP.

Generally, IP spoofing attack can be implemented in two ways to overload targets with traffic. One method is to simply flood a selected target with packets from multiple spoofed addresses. This method works by directly sending a victim more data than it can handle. The other method is to spoof the target's IP address from distributed resources and each of resource send packets from that address to many different recipients on the network. When another machine receives a packet, it will automatically transmit a packet to the sender in response. Since the spoofed packets appear to be sent from the target's IP address, all responses to the spoofed packets will be sent to the target's IP address, this is known as Distributed Denial-of-service(DDoS). IP address spoofing attack also have others implement rather than DoS, such as Man in the Middle

2.3.4 Man in the Middle

A man-in-the-middle attack (MitM) is an attack where the attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other. The attack is a type of eavesdropping in which the entire conversation is controlled by the attacker. MitM attacks pose a serious threat to online security because they give the attacker the ability to capture and manipulate sensitive information in real-time, IP-spoofing was considered as the first step toward a working man-in-the-middle attack. In these attacks, a malicious party intercepts a legitimate communication between two friendly parties. The malicious host then controls the flow of communication and can eliminate or alter the information sent by one of the original participants without the knowledge of either the original sender or the recipient. In this way, an attacker can fool a victim into disclosing confidential information by 'spoofing' the identity of the original sender, who is presumably trusted by the recipient.

2.3.5 Denial-of-Service Attack

IP spoofing is frequently used in DoS which is currently one of the most difficult attacks to defend. Because hackers are care only with consuming bandwidth and

resources, they need not concern about properly completing handshakes and transactions. Rather, they wish to flood the victim with as many packets as possible in a short amount of time. When multiple compromised hosts are participating in the attack, all sending spoofed traffic, it becomes a real challenge to block traffic. TCP/SYN Flooding is one of the most typical attack in DoS, in this attack, an attacker sends TCP SYN packets as if to initiate a TCP connection with its victim. These SYN packets contain spoofed source IP addresses, which cause the victim to waste resources that are allocated to half-open TCP connections which will never be completed by the attacker

3 Web session authentication

In this section, we gave a brief introduction of what is HTTP cookie, including the property and functions of HTTP cookies. And we introduced what are web session authentication and session cookies, and how session cookies contribute in web session authentication. Beside these, We also introduced the current threads for cookies, among these threads, we focused on a specific thread, that is called session hijacking.

3.1 HTTP cookies

Web session typically refers to a continuous period of time of interactive web information exchange between two parties(normally refer the web browser and web server), For example, from the perspective of web application, a web user open a e-commence website by a web browser, login it, and do some transactions like buy something from the e-commence website, and then logout, the whole web operations will be treated as a web session. if from the perspective of web application developers, the web session more like a data structure that used to store users' information such as user username and password for login, or user profiles. no matter from which perspective, a web session needs to refer its context, in other words, a web session needs to record something in some where as it works based on some previous information. However, the traditional HTTP protocol is a stateless, which means it has no ability to record some actions which have done before to main the connection state. This design is consistent with the original purpose of HTTP protocol. The original purposes is to design a protocol that can publish, parse and receive HTML tags. In early time, there is no dynamic web page technologies, all web pages only

include static HTML scripts, so web browsers only need to simply send requests to web servers for downloading static resources, so there is no need for web browsers and web servers to record connection state. Beside this, when a web user receive a response from web server, most of time he will spend a relatively long time to read this response before he makes the next request to web server, so always keep the connection on is a waste of resources, and do not put state maintain ability makes HTTP protocol simple, and we can include other advanced ability on other protocols which based on HTTP protocol, thus can make HTTP protocol having better scalability.

However, as the time pass and the development of web technologies, the dull static HTML webpages can not meet the need for people's network activities any more. For example, people begin to move previous offline business transactions to online, the webpages begin to more and more complicated and sophisticated, and start to have the property of interaction, and each request and response for a web user between web browser and web server can not be treated as an independent transaction any more. The need for web session management is increasingly important and urgent. In order to deal with this need and give HTTP protocol the web session management ability. an important and simple technology, cookie, has been proposed. In 1994, cookie is first been proposed by Netscape for web session management. As the highlights of its efficiency and simple to deployment, it is rapidly adopted by major web browser and soon to become the default mechanism for web session authentication. Beside cookie, an other web solution to maintain state between web browsers and web servers is session.

3.2 Difference between cookie and session

Before we introduce what is cookie and its mechanism, we will first give some examples to show the difference of mechanism of cookie and session. The example is based on the scenario that there is a coffee shop, the shop has a discount plan that any consumer who buy up to five cups of coffee will be rewarded one free cup, however, most of time it is hardly possible for a consumer to buy five cups of coffee in one time, so there is necessary to have a mechanism to keep track of consumer's consumption. The below shows the possible plans for recording consumer's coffee consumption.

Plan 1. The coffee shop assistant has a powerful memory ability that enough to keep every consumer's coffee consumption in mind, so by this plan, there is no necessary

to design any extra plans, and just need to keep all consumption record in mind, as long as consumers walk into the shop and buy coffee, the shop assistant do know what to do with this consumer. If we compare this way in HTTP protocol, using mind to keep all previous consumption record corresponds to make HTTP protocol itself to support state maintain.

Plan 2. The plan 2 is to give every consumer a consumption card, which records the number of cups of coffee the consumer has bought so far, and there will also have a validate period on this card. When a consumer buy a cup of coffee, he can show this card to shop assistant, and consumption will be associated with previous consumption(the number of cups of coffee bought will increase one). If compare in HTTP protocol, this way is like to make client side to keep state.

Plan 3. The plan 3 issue every consumer a membership card, different with consumption card in plan 2, apart from a membership serial number, the membership card does not record any information related to the consumer's consumption. Everytime when the consumer begin to buy coffee, he only needs to show the membership serial number in the membership card, and the shop assistant can find consumption record in computer by entering the consumer's membership number, then can add one new consumption record on this consumer's consumption list. if compare in HTTP protocol, this way is like to store state record on server side.

Due to some historical reasons, HTTP protocol is designed to stateless and better not to change to state in the future, thus the latter two plans become the options. The cookie mechanism corresponds to the plan 2 that used to keep state on client side, and session corresponds to the plan 3 to keep state on server side. And as keep state on server side also need a identifier to be kept on client side, so in practice, the session mechanism also needs cookie to keep the identifier on client side, thus to achieve the final purpose.

3.3 Cookie mechanism

As described before, the cookie is like the consumption card that can record consumption record in consumer side, actually, cookie a piece of data structure which stores session information and is stored in client side. The cookie generation is achieved by using HTTP protocol, the web server will instruct web browser to generate cookie by adding some instructions by using set-cookie in HTTP response header field, some client scripts like javascript or VB script can also generate cookie.

The use of cookie is the web browser according to some certain principles to send cookie to web server. Everytime when web browser try to send requests to web server, the browser will check all stored cookies, if the scope of one declared cookie greater than or equal to the requested resource location, the web browser will add this cookie to the request and send to web server.

Cookie consists in a form of name-value pairs. The fields of one cookie include a part called cookie name and cookie value,the name and value are used to store session information. For example like Cookie: session id = 8d6fegdfgf8fgd4.Cookie also include an expiration date,the expiration date indicates the lifecycle of the cookie, in other words, how long the session information can be kept by the cookie. If web users do not set expiration date for the cookie, the lifecycle of this cookie is equal to the session period of the browser, once the web browser is closed, the cookie is destroyed. usually we called this kind of cookie as session cookie. generally speaking, session cookie is stored in web browser instead of hard disk. however if the expiration date is set, the cookie is stored in hard disk and can be shared by different browsers, and the cookie will is still valid even the web browser is closed until reaching the expiration date. It also keep domain information which we can assign a domain in this field like google.com, and also can assign a specific host under this domain like www.google.com or froogle.google.com and a path which is appended to domain, it is a section of url like / or /foo. The domain and path together to form the scope of cookie.

3.4 The functions of Cookie

Usually cookie has three main uses.

- Web session management

  Now most of websites have login functions, once a user successfully login the website, then website need to main the user session in a friendly way. However the HTTP protocol is stateless and does not support session management. The most used authentication way is Base Auth, however, by using this way, the private information like username and password will be transmitted in a plain way in network, and suffers from network attacks like man-in-the-middle. Now most of websites solve this problem by adopting cookie to maintain session, when a user successfully login the website, a unique identifier will be generated and stored in cookie, every time when web browser send request, the identifier

will be sent with the request as the identify of web user, the web server will authorize the access to the user based on the unique identifier.

- Customization

  Cookie can also be used to record some data for customization. For example, iGoogle(a service provide by google and now has be closed) can allow users to customize their home pages, and this product just used cookie to record user preferences.

- User tracking

  cookie can also be used to track user behaviors, like whether a user visited the website or not, or what operations the user did when visiting the website.

3.5 Session mechanism

An other way to maintain session state between web browser and web server is session. Different from cookie, session is stored in server side. when a web application need to create a session for request, the web server will first check if the request already contains a session identifier, we usually call it session id, if so , that means web server has already created a session for the user, and will extract the session according to the session id. If the request does not contains a session id, the web server will create a session and a session id which is associated with the session, the session id will be sent back to web browser in HTTP response header. For storing session id, the common practice is to store session id by cookie, and in later communication, the session id can be sent along with request to web server.

3.6 The comparison between cookie and session

Although cookie and session are all used to record information, they have some difference in usage scenario, security and validation period aspects.

- Usage scenario

  Typical usage scenario for cookie is "remember me" function, the user account information can be stored in cookie, when the user request for login for the same website, the user account information can be extracted from cookie and sent to web server for automatic login, and avoid user to type username and

password again.For session, the typical usage scenario is after user login the website, the logon information such as session id can be put in session, and these logon information can be verified for each later request to ensure the legitimization of the request user.

- Security

  The web session information can be either store in cookie in client side or in session in server side, however, if session information like username and password are stored in cookie, for every request these sensitive information will be sent over network, thus to impose the risk of being stolen, so the common practice is to put session information in session in server side, and only store identifier like session id in cookie.

  Cookie can also be used to record some data for customization. For example, iGoogle(a service provide by google and now has be closed) can allow users to customize their home pages, and this product just used cookie to record user preferences.

- Validation period

  As cookie's expiration period can be manually set to a long time, so the validation time for cookie varies from minutes to even months. However, validation period for session is usually short, usually the session will be expired by user logout operations or closing webpages.

3.7 The session hijacking thread

Although cookie is useful for web session management, however, it is suffers from risk of being stolen. As cookie maybe stored in web browser, some attacks can make use of this feature like XSS(cross site scripting) to steal cookie. According the above description, we can set the HttpOnly flag of cookie, in that way client side script can not access to cookie any more. Another limitation is cookie is static, and it will not change in its lifecycle until reaching the expired time, and it is transmitted unprotected over the network, so adversaries can use some tools to steal cookie when it is passed over network, and can reuse this session id in cookie to impersonate the victim to access the session. We called this attack as session hijacking.

Figure 2[DCA11] shows the whole flow of session hijacking attack. After finishing user's primary authentication, the session cookie is generated. the victim send

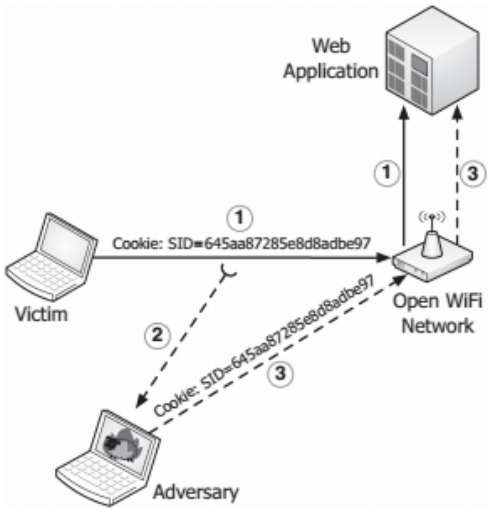

Figure 2: Session hijacking attack

its request along with the cookie(SID) as auhentication token to web server for authentication. As cookie is passed unprotected over the network, the adversary can use some tools like FirSheep to capture the victim's cookie. After that, as the cookie is static and unchanged during its lifecycle until reaching the expired time, the adversary then can impersonate the victim by using the victim's cookie to make arbitrary requests to web server. It is important to note that even the victim logout, the cookie is still validated until the it is expired.

In the below, we will discuss some current solutions for preventing against session hijacking and their strong points and limitations.

4 Current solution for session hijacking

The security thread of using cookies for client authentication and session management has raise concern since it was first proposed. In the below, we will discuss some works so far has proposed to protect against session cookie thread, especially for session cookie hijacking.

4.1 HTTPS

a typical, and most popular way that has been adopted most today to protect against session hijacking is by using HTTPS protocol, which is a secure version of HTTP protocol, and by combining HTTP(Hypertext Transfer Protocol) with the SSL/TLS(Secure Sockets Layer/Transport Layer Security) protocols to provide encrypted and authenticated communication to web users.

We used HTTP protocol for web communication for a long time before HTTPS is first proposed. HTTP is short for Hypertext Transfer Protocol. HTTP is first proposed for the use of World-Wide-Web(WWW) since 1990, it is somewhat misleaded by people as a protocol for transferring hypertext, is actually a protocol for transmit information, the data transmitted by HTTP plain text, images, audio, video or any type of Internet resource, that is why it is called hypertext. In addition to the content it transfer, HTTP also has the following several features:

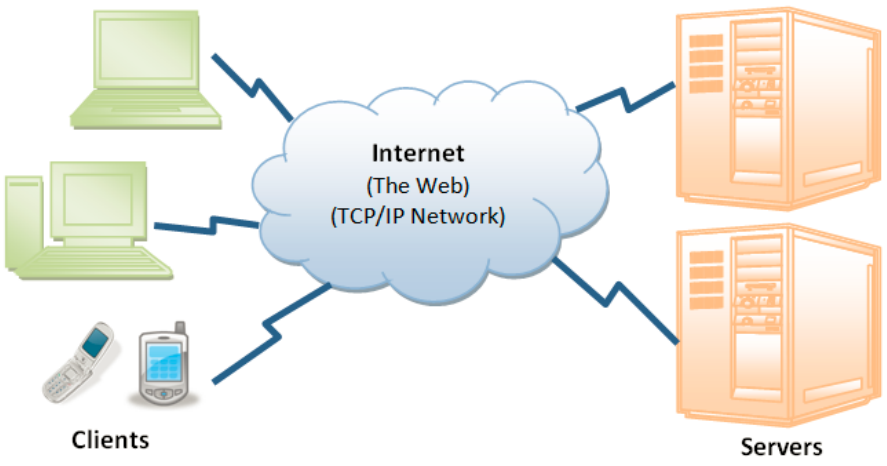

Figure 3: Client/Servers structure

- Client/Servers structure

  As shown in Figure 3, HTTP protocol is a client/server based protocol, a typical usage scenario is between a web browser and a web server. However, as the improvement of web and network technology, now sometimes the data we intend to access over we browser may reside on several servers instead of just

one, so the web administrators have to concern on the state synchronization for some large web application reside across distributed servers.

Typical usage scenario for cookie is "remember me" function, the user account information can be stored in cookie, when the user request for login for the same website, the user account information can be extracted from cookie and sent to web server for automatic login, and avoid user to type username and password again.For session, the typical usage scenario is after user login the website, the logon information such as session id can be put in session, and these logon information can be verified for each later request to ensure the legitimization of the request user.

- A pull protocol

  HTTP is a request-response type communication protocol. A typical HTTP transaction processing is an web client send an HTTP request to web server, the web server parse this request, prepare the content the request asked, like text, images, videos from the server document base directory, and send back to web client, or execute a asked program, send back the output to web client. So HTTP protocol is also called a pull protocol, the client pulls information from web server instead of server push data to web client.

- Application layer protocol

  Among the ISO 7 layer model, HTTP protocol is resided on the application layer. HTTP protocol is upon on TCP protocol, and it makes use of TCP protocol. Every time when user send a HTTP request to web server, HTTP will use TCP as a vehicle to transmit HTTP packets, and will also create a TCP connection between web browser and web server. The TCP connection will terminate as soon as the communication is completed, so HTTP protocol inherit some features of TCP protocol.

- Connetionless and stateless

  However, different from TCP protocol, HTTP protocol is a connectionless and stateless protocol. The connectionless means every time the web server can handle only one request, the web sever will terminate the connect with web client as soon as the request handle has completed by server and has received response from client. HTTP is designed to as this way is because at early time, web servers may handle millions of requests from all over the world, however, the interval between each request is relative long, if HTTP is designed to keep

the connection alive along the entire session, most of time the connection is just alive instead of doing any thine, and it may waste time and network resource. So at that time, HTTP is designed to establish connection when the web client send request, and terminate the connection immediately when the request is handled by web server. However, as the time pass, the web pages become more and more complex and contains more and more hypertext data like images, it is very inefficient to establish every connection for every image access, so later, the Keep-Alive function has been proposed to address this problem. Keep-Alive enable the persistent connection between web client and web server, and can avoid establish or reestablish for the subsequent upcoming request to the same server. However, it also has its limitation, as Keep-Alive keep the connect alive, it will keep some resource taken which may be released after access if without Keep-Alive function, thus may effect the performance, the impact of the utilization that Keep-Alive to resource is especially significant when web server and we application run on a same machine.

The stateless is a very important feature to HTTP, and lots of other web concepts like cookie and session are associated with the stateless. The stateless means HTTP protocol can not precept and maintain the previous session state, in other words, HTTP protocol has no memory ability for HTTP transaction processing, the web server does not know what the state of client is. When web client send HTTP request to web server, the web server process the request and send back response to client, after that, not any record for this request processing or client information will be kept in server side, this means each request is independent with each other, and Keep-Alive feature can not also solve this problem. The stateless feature gives some benefit to HTTP, at the same time, it also has some limitations. As web server does not need to memorize and keep state and other information for web client, it free web server avoid unnecessary resource taken, and also can increase the response time. However, as web server has no information about the requested client, if subsequent processing need previous sent information, the web browser must retransmit the previous data, this may increase the amount of data that transmitted for every time you connect.

As the time pass, the web application has become more and more interactive, and the stateless feature will impede the interaction as interaction need context information, like shopping cart, the web site needs to know what commodities the user has selected previously. Under this situation, cookie and session, the two technologies

to keep connection status, were proposed, cookie can keep login information to the next session that the user act with the same web server, so the user does not need to input username and password every time. Compared to session, cookie is stored on client side. A typical usage scenario is to determine if a registered user is already login on the website or not. Users may be prompted to tell if they willing to retain user information by cookie to simplify the next logon procedure at the first login time, they are all cookies' function. Besides cookie, session is another solution for keeping connection status. Session is stored on server side. When client access server, the server will setup the session and store session information on the server, and also generate a session id and pass to client, the session id is normally stored in cookie, so after that, every time when client send request to server, it will also send the extra parameter session id to server, and server can retrieve session information according to the session id. Even if the client browser shuts down unexpectedly, the session information is still keep on server, so as long as you know the session id, you can still continue to request information from this session, and some attack has taken use of this feature to steal uses' private information. The session can be set a session time-out, once it exceed the session time-out time, the server will clear the session information, by using this way can prevent session stolen at some extent.

4.2 URL

URL, which is also called Uniform Resource Locator, is used by HTTP protocol to describe and identify the location of a resource on the Internet. A formal URL is consists of four syntax just like the following: *protocol://hostname:port/path-and-file-name*

protocol syntax tells browser use which protocol to access the Internet resource. The most common protocol is HTTP protocol,hostname tell browser which host or machine the resource resides on, it is DNS address or IP address of the machine.Port tells browser which port the server is listening to the coming requests. and path tell the path of the requested resource under server document base directory. Although URL is not designed for HTTP. It is a very important part for HTTP, even some research like in SessonLock [Adi08] has make use of the url fragment identifier to prevent against session hijacking.

4.3 HTTPS

As HTTP transmit data in a plaint way, the data is easily captured or intercepted by other attackers, make the communication insecurity, so HTTPS is proposed to tackle this problem. As mentioned before, we know that among OSI 7 layer model, HTTP is on the top layer upon on the TCP protocol, and based on the TCP protocol, the HTTPS protocol integrate another secure layer with two secure protocols, the SSL(Secure Socket Layer) or TLS(Transport Layer Security) between HTTP protocol and TCP protocol, so HTTPS is a communication protocol which combine HTTP protocol and SSL/TLS secure protocols to provide a encrypted communication and web server identification.

HTTPS protocol involves two important security elements, that are web server authentication and secure communication. The purpose of web server authentication is by verify the certificate of web server against the certificate of CA(Certificate Authority) to ensure the identity of web server, thus to ensure the security communication. The workflow of web server authentication can be seen in Figure 4[Her16]. There are about three steps involve web server authentication process.

- The first step is to install CA root certificate. The CA will distribute the CA root certificate to browser, and browser distribute it as part of its installation.
- The second step register certificate from CA. During this step, web server need to register its certificate from CA by sending request to CA, the CA will work as a RA(registration authority) to verity the identify of web server, once successful verity, the CA will issue a signed certificate which contains public key to web server for up coming authentication and encryption.
- The third step is authenticating web server. After the second step, web server has got its own certificate, then when the browser want to access the web server, it will first receive the certificate from web server, and then work as a VA(Validation authority) to validate the web server certificate against CA root certificate, if the authentication is successful, thus we can ensure the trusty of web server, then browser can use the public key contained in web server certificate to secure its communication channel with the web server.

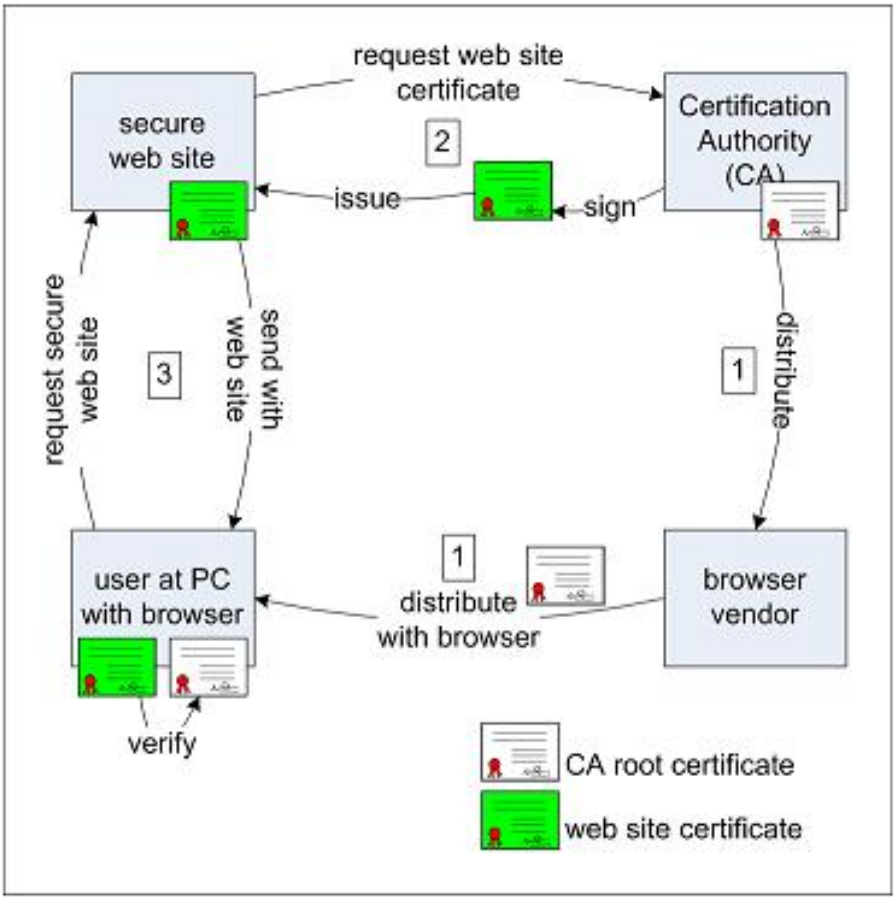

Figure 4: Web server authentication

4.4 Secure communication

After web server authentication, we can trust the identify of the web server that we plan to visit, then we need to secure our communication channel to prevent passive or positive attacks such as main-in-the-middle or eavesdropping. According to below introduction we know that HTTPS use two secure protocol, the SSL and TLS, actually both of the two secure protocol involve using a PKI(public key infrastructure) system, this system is a asymmetric system and contains two type of keys, the public key and the private key, the public key is used to encrypt things, and the encrypted content can only be decrypted by the private key, so the private key must be kept security, and the public key can be distributed to anyone who intend to encrypt things. The two keys are used to securely delivery a encryption key,

which is used by HTTPS to encrypt content that browser send to of back from web server. So the secure communication step contain two major steps, the encryption key delivery and data encryption.Figure 5 shows the secure communication flow.

Figure 5: Secure communication

4.4.1 Encryption key delivery

1. After authenticating web server, the browser got the certificate from web server which contains a key public that is used to delivery encryption key. Then browser generate the encryption key by using a secret key algorithm.

2. After generated the key, the browser uses the public key encrypt the encryption key, and send over the Internet to web server.

3. After web server receive the encrypted encryption key, it uses its private key to decrypt it, and then get the encryption key, after that, encryption key delivery is finished

4.4.2 Data encryption

1. The browser encrypt the data it want to send by using the encryption key.

2. The browser then send the encrypted data over Internet to web server

3. The web server receive the encrypted data, and use the same encryption key to decrypt it can get the original data
4. The web server encrypt the data that will back to browser by using the encryption key.
5. The web server send the encrypted data over Internet to browser.
6. The browser receive the encrypted data and decrypt it by using the encryption key, and then get the original data.

According the phrases we can see that by adopting this way, we can protect data from leakage or being hacked by other attackers as the data is encrypted and can not be decrypted without private key. However, at the same time we notice that this way involve heavy computing cost and may affect the performance of transmitting data.

4.5 HTTPS protect against hijacking

As traditional HTTP protocol exchange payload between web browser and web server in a plain way, during web browser and web server interaction, the session cookie is under insecure situation and is easily to be listened or intercepted by other attackers, so the most straight way is to directly encrypt the communication data between web browser and web server, and HTTPS just provide this encryption service. Now lots of website or services like Gmail, Hotmail, Facebook and Google are adopt HTTPS to secure their communication.

4.5.1 Performance impact

One drawback of adopting HTTPS protocol to protect web application against session hijacking is it will cause significant performance impact. Deploy HTTPS over the while web site will trigger numerous computations on server side, even if we adapt high computational power machine on server side, the computational cost is still considered expensive, and it also will trigger much work on client side to affect the performance, thus to affect the user experience. Some studies have shown the performance impact that HTTPS on web sites, for example in [CDW06], the author conducted a series of experiments to show how TLS protocol effect the performance of web sites. The experiments all contains six sets of experiments, these

experiments used two different workload traces with three different machine configurations to compare. One workload trace comes from Amazon with average file size of 7KB, and total set size of 279kb. The other trace is departmental trace which the average file size is 46KB and the total set size is 530MB. Normally the amazon web handle user interactive behaviors through normal web server and put private operations like credit card information and personal information via secure web server.

By contrast, the department web server is under normal and unencrypted circumstance, and the experiment replace the normal web server by equipped it with TLS protocol, and to measure the throughout of traces under TLS protocol. By evaluate the results of experiments show that TLS protocol has a significant effect on performance of web site by imposing a factor of 3.4 to 9 times cost than an normal and non-TLS protocol web site, the factor that contribute most for cost in TLS protocol is the public key cryptography, and it takes 13% to 58% of computational cost.

For non-TLS parts, the cost take 19% to 45% of the total cost. From the results we can know that deploy HTTPS has a significant impact on the performance for the web sites. In addition, HTTP will also affect the compatibility of web sites, like the cache behavior for web browser will be different under adoption of HTTPS, and will increase the page load time. In order to solve the heavy computation problem, the most common way is to just deploy the HTTPS protocol for these which need to be protected, and the most common scenario is user login and authentication phrase, and leave other unimportant pages over traditional HTTP. When the user login authentication phrase finished, the page will automatic switch from HTTPS to traditional HTTP to void encrypt these unimportant things, thus to decrease the negative effect on performance that HTTPS imposed on web site and to help web servers get rid of heavy computation. However, by this way, although the user private credential(normally the password) is encrypted and protected, the subsequent requests are still in plain and exposed to attackers, and these privacy still suffer from the risk of being stolen. Apart from narrowing the usage scope of HTTPS, some companies choose some acceleration hardware or special hardware for handle HTTPS server side heavy computational overhead, however this way will increase the cost for these companies.

As direct deploy HTTPS will have negative performance impact, and may cause compatibility problem for web browsers, some researches try to circumvent this by integrating encryption with other protocols, for example in [BHH+10], the author

proposed a new protocol, the tcpcrypt, which integrates encryption with TCP protocol to make the encryption happen in transport level. The tcpcrypt has several significant progresses compared to HTTPS or VPN, firstly, different from normal secure protocols like SSL or TLS, tcpcrypt needs no configuration, so it is easy to deploy, and have no affect(normally refer to change or modification) for existed applications, so there is no compatibility problem, and high robust ability, even if the remote end does not support tcpcrypt protocol, the network connection can continue work well. Secondly, tcpcrypt supports gradual and flexible deployment, it can easily switch bwtween tcpcrypt and traditional TCP protocol. The Internet traffic will be encrypted if the other end that you communicate with tells that it support tcpcrypt, otherwise, the communication channel will be in plain mode. Thirdly, according to the result of experiments that the author conducted, it revealed that tcpcrypt has a significant improvement on performance compared to HTTPS, it is 25 times faster than SSL, so if suitable for many type of servers and many type of situations especially need heavy computations. Lastly, the current solutions include HTTPS and VPN, are not inadequate as they all rely on PKI(Public Key Infrastructure) for authentication and prevention, so all network parties using the protocol to communication need to verify their identities from CA(Certificate Authority), and have to buy certificate from CA, however, tcpcrypt enable ubiquitous deployment and support any authentication mechanism include PKI, password or others.

However, there are still some limitations prevent it from large scale deployment. Firstly, we divide network attack into two types, that are passive attack and active attack. The passive attacks actually are not real attacks, the attackers are normally the unauthorized parties that listen to the network, or infiltrate the network for later active attack. The main purpose of passive attack is to gather network traffic on the target, and often the preparatory activities for active attack, for example like intercepting data of the target to help attackers for preparing an active attack for this target. The common passive attacks include eavesdropping. By contrast to passive attack is the active attack, they are the real attack and attempt to listen network and intercept data on a target, modifying the gathered data and replaying to the target. The most common active attacks like MITM(main in the middle) attack. For tcpcrypt, the passive attacks are much simpler to prevent as they just listen and monitor the network, if the network traffic are being encrypted, the passive attackers can do nothing, however, for active attacks, tcpcrypt is vulnerable and can not provide guarantee to this attacks, because if active attackers modify the response data to impersonate server and tell client tcpcrypt is not supported by

server(actually it is supported), the subsequent communication will be plain way and let the communication under risk of leaking. Secondly, tcpscypt just provides best effort instead of guarantee, so the tcpcrypt can not be used alone and should work with other protocols to ensure the security.

4.5.2 HTTPS limitations

the second drawback of deploying HTTPS as solution for preventing session cookie hijacking is sometimes even with HTTPS enabled, there still have some ways that can be exploited to steal session cookies. The HTTPS protocol ideally can be used to protect web sites from being attacked including passive attack and active attack, however in real world, lots of web sites are willing to deploy HTTPS in a incorrect way for reasons, for example, in order to meet the compatibility that the web site with HTTPS, most web sites choose to compromise at the price of security by configuring HTTPS in a incorrect way. And some website owners who are less-security-awareness may think that the incorrect configuration of HTTPS is enough to protect against passive attack, and the cost of deploy correct HTTPS outweighs the risk of active attacks. Apart from these, most of users lack awareness of network security, and tend to ignore some mistakes of security configuration like HTTPS indicators absence or warning pages, and some previous studies has shown these. For example, In [SDOF07] the author designed and conducted some experiments to evaluate how users behave at the situation of HTTPS indicators are absent, the site-authentications are absent and the warning pages are displayed when they are asked to input passwords. The experiment asked 67 participants to conduct a common task that login their bank accounts for three times, with each time of logging in, the experiments will gave them increasingly alarming clues that indicate the connection was insecure, and to see how these participants response to these insecure alarms. These participant were divided into three groups, the first group were instructed to play a role, the second group were gave some additional instructions, and asked to use the same role-playing scenario, and the third group were asked to login by using their own bank accounts. During the first round of login, the HTTPS indicators was removed, however, still 63 out of 67 participants entered their password and complete the task, during the second round, the site-authentication images were removed, 58 out of 60 participants were still entered the password, despite the clues indicate that that the connections were no longer security. During the third round, with the warning pages displayed, still 30 out of 57 participants entered their

password. The result of experiments revealed that most of people lack the awareness of information security, and tend to ignore the security indicators or alarms when operating some privacy, even under the situation of absence of HTTPS indicator or site-authentication images. More importantly, even with the presence of s site-authentication image does not guarantee the security of the connection, not mention to the safety of entering a password. So the HTTPS can not guarantee the ultimate safety against attacks like session cookie hijacking.

As most people lack awareness of network security and tend to ignore the mistakes of security configuration, in paper [JB08], the authors, Jackson and Barth, has proposed a new security mechanism, called ForceHTTPS. They implemented it as a browser add-on, the ForceHTTPS can protect websites from insecurity connections, force to ask browsers to treat each HTTPS errors or configuration mistakes as attacks, not just some insignificant configuration mistakes. By enable the proposed ForceHTTPS, the browser will follow the below security guidelines:

1. All non-HTTPS connection will be treated as insecurity connections and will be automatically switch to HTTPS connection by web browser, thus to ensure encryption of communication traffic, and prevent some privacy like session cookies directly expose to potential attackers

2. Some TLS errors which are allowed previously without ForceHTTPS enabled are forbidden, like self-signed certificates, and will case the termination of TLS session.

3. Attempt to embed some insecure content into web pages like scripts, CSS(cascading style sheets) will be forbidden by ForceHTTPS, thus to prevent some potential attacks that may make use of bugs of scripts to lunch some active attacks like script injection.

By using ForceHTTPS, websites can get benefits from several aspects. Firstly, ForceHTTPS conduct strict error handling that to help web site get rid of insecure communication. As ForceHTTPS will automatically switch insecurity HTTP communication to HTTPS communication, can encrypt communication channel thus to security our privacy like session cookies. Secondly, ForceHTTPS can block websites away from malicious and insecure contents like scripts and CSS, and can prevent some script attacks. And as some mistakes or errors of HTTPS are treated as attack by ForceHTTPS, thus force website owners to correct the configuration of HTTPS,

and can enhance the security level of websites. ForceHTTPS is especially suitable for some high-security-conscious websites that can not allow any mistakes or risks such as bank system.

Although there are lots of highlight of ForceHTTPS, some limitations from its nature still prevent it from large scale deployment.

1. One of limitations is come from the nature of ForceHTTPS that the enable of ForceHTTPS is established up on the first secure connection of web browser to web server. As the ForceHTTPS cookie has to be set during the first secure connection, if the web browser can not establish the first connection in a secure way, the ForceHTTPS cookie can not be set, and the security mechanism can not be executed. If a attacker can control every visit of the web browser to the web sever, thus can prevent the ForceHTTPS to be enabled, under that situation, even warning messages are able to show to users, the ForceHTTPS can not do any security operations. By contrast, once the first secure connection is established and the ForceHTTPS is enabled, the ForceHTTPS will conduct its security guidelines until the ForceHTTPS is expired.

2. The second limitation is the protection of ForceHTTPS to website associated with the presence of ForceHTTPS cookies. Like other cookie, the ForceHTTPS cookie is also stored in browser and waited for other ForceHTTPS websites to use. If one user delete the ForceHTTPS cookie, the protection of ForceHTTPS for the website is also removed.

3. Although ForceHTTPS can automatically correct the errors of mistakes of HTTPS configuration, and let the web communication is under secure, however, it can do nothing for some other attacks.

4.5.3 XSS(cross-site scripting)

ForceHTTPS can not provide protection the website if the website contains cross-site scripting vulnerability. This a common vulnerability for most of website, according to the statistic data issued by symantec in 2007 [DTM08], XSS vulnerability carry out roughly 84

4.5.4 CSRF(cross-site request forgery)

Similar to XSS, ForceHTTPS is not also suitable for protect websites which contain cross-site request forgery vulnerabilities. Different from XSS that by injecting malicious script to web pages to execute command, CSRF exploit the trust of a authorized user to a website to execute a unauthorized commend to that website. The fig5 describe the general workflow of CSRF. There are two main steps for lunch CSFR attack, generate trusted cookie and exploit the trusted cookie to access the website in an unauthorized way. For first step, the victim visit the websiteA which has the vulnerabilities of CSRF, then generate the trusted cookie for website A in browser. During the second step, when the victim visit the website B under the situation that he didn't logout from website A and the trusted cookie is not expired, the website B can forge the request that ask for the visit of website A, then according the request, the browser will use the trusted cookie to visit website A, in that way the website B can visit website A although it has no permission to visit. For this attack, ForceHTTPS is still can do nothing

Based on this work, a similar security mechanism called HTTP Strict Transport Security(HSTS) has been proposed by IETF[HJB10]. It can allow web servers to tell that they can be only accessed by web browsers in a secure way(by using HTTPS secure protocol instead of HTTP). When one user enter a url in web browser without explicit point out the protocol, HSTS can automatically switch it the HTTPS secure protocol. For example, when you type google.com, the browser will automatically forward to visit https://google.com instead of http://google.com.

An other similar approach that rewrite the requests to ensure web pages being visited over HTTPS has been developed by the tor project and Electronic Frontier Foundation(EFF), and it's name is HTTPS Everywhere. HTTPS Everywhere is a free and open source browser extension, and it's compatible for most main stream browsers like Chrome, Firefox, Opera. Similar to ForceHTTPS, HTTPS Everywhere can rewrite the requests to make sure the websites use secure HTTPS connection instead of HTTP, thus to ensure the information security.

However, according to the studies in [CMWZ09], both ForceHTTPS and HTTPS Everywhere have limited protection ability on active attacks such as MITM(man in the middle) attack. This paper is motivated by the curiosity that whether the integration of HTTPS protocol into web browser can really prevent against the same adversary that is considered in the design of HTTPS protocol, and whether the HTTPS can really defend all kinds of attacks and protect web communication

from being intercepting and attacking. The author focused on an adversary called "Pretty-Bad-Proxy" (PBP), the PBP is actually a malicious proxy that conduct main-in-the-middle attack to break the end-to-end security mechanism of HTTPS thus attempt to access and modify sensitive data. The experimental situation is that, the web browser communicate with the web server in a secure way by using HTTPS protocol, so the communication channel is encrypted. The target of PBP is the rendering modules of browser which lays above the HTTP/HTTPS layer, the purpose of PBP is to send malicious data to browser's rendering modules through unencrypted channel, and thus try to access or forge private data or get access right. The experiments assumed that PBP can not encrypt or decrypt data which transmit on the Internet, but can access the raw and unencrypted data on browser. As HTTPS protocol is designed to provide web communication security service and prevent websites from passive or active attacks such as eavesdropping or man-in-the-middle, so in this paper, the author list three assumptions and made experiments to verify whether the three assumptions are correct or not. The three assumptions are:

1. As HTTPS protocol provide secure communication service, so the communication content on the Internet between web browser and web server is encrypted and safety, and can not be stolen by malicious hosts.

2. As HTTPS provide server authentication service, so the identity of server can be authenticated and store in a form of certificate.

3. There is no malicious host can impersonate authenticated server to access servers.

Based on these assumptions, the author conducts two kind of experiments, the script-based vulnerability experiments and static-html-based vulnerability experiments. The scripting is becoming a increasingly important capability in web development, it not only share some computational burden with servers, and can execute some tasks that can provide users much better experience. However, the powerfully capability also impose some risks about web security, the most well-known scripting attacks are cross-site scripting and browser cross-domain attacks [CRW07]. In the script-based vulnerability experiments, the author conducted three script-based attacks to verify whether the HTTPS protocol can protect websites from these scripting attacks. The three attack experiments are by embedding scripts in error message, redirecting script request to HTTPS websites, and imposing scripts into contexts

via HPIHSL pages. In the static-html-based vulnerability, the author make use of static HTML contents to conducts two attacks to verify the vulnerabilities of HTTPS. The results of these experiments showed that the PBP indeed can exploit some vulnerabilities of HTTPS protocol, thus to indicated that HTTPS protocol is no the ultimate solution for secure web communication, and still has some security flaws that can be exploited , for example form scripting, from static HTML, from HTTP proxy, to break HTTPS security mechanism and to obtain sensitive data from servers that under protection of HTTPS.

From the discussion above we can conclude that HTTPS is a very popular and relatively useful way so far to protect website against network attacks, it combine HTTP protocol with two type of secure protocols that are SSL and TLS, to provide traditional HTTP protocol with the security ability, like server authentication, with this authentication, web server can prove its identity by the issued certificate from CA, and web browser can trust the certificate, thus to distinguish trusted web server from malicious web servers. And the ability of secure communication. This ability equip HTTPS protocol to encrypt the communication channel, thus to ensure the confidential of communication content, especially for some important content like session cookie, so network attackers can not steal the session cookie, thus prevent attackers to impersonate the victim to the session with the server. Although these powerful abilities can ensure the security of websites at some extent, we can not ignore that there still have some limitations and flaws of HTTPS protocol. One very apparent limitation is deploy HTTPS protocol will trigger numerous computations on server side, and also cause more work on client side. And HTTPS protocol itself can not also guarantee one hundred percent security as some limitations, no matter from users' behavior like normal users tend to ignore errors or mistakes of HTTPS configuration, , or from the flaws of HTTPS protocol itself, for example, by some previous experiments shows that HTTPS can not provide protection for some attacks like script-based attacks or static-html-based attacks. And although some solutions have proposed to solve these limitations, like the proposed ForceHTTPS, the new security mechanism HSTS, or the browser extension of HTTPS Everywhere, they all have some flaws that prevent them from large scale deployment.

As the limitations and flaws of HTTPS, some research work has focused on very different directions, and some new solutions that completely different from HTTPS have been proposed.

4.6 One-time cookies

One innovative way that proposed is called one-time cookies(OTC)[DCA11]. As we all know that, session cookie is played as authentication token that generated by web server after authentication of web user, and in later communication, the authentication token is used as a unique credential for authenticating the identity of web user, thus avoiding web users to input password to login explicitly for every subsequential communication with web server. However, in HTTP protocol, the session cookie is transmitted on the Internet in a plain way, and the traditional session cookie is static and never change during its life circle until the expire of session cookies, so it suffers from the risk of being stolen, and even can be made use of by attackers to impersonate as victims to gain the access to the session with target web servers without authentication. Although we have HTTPS, the security version of HTTP, can provide encryption service to ensure the confidential of communication channel, however as some limitations or flaws of HTTPS, like the negative impact on performance of websites, the compatibility of web browsers, most companies just deployed HTTPS for user login authentication, and leave other web pages by using normal HTTP, and this way is not enough to ensure the security of session cookies and prevent from session cookie hijacking attack.

The OTC protocol, is proposed to overcome the limitations of HTTPS. The highlights of OTC protocol is by designing a hash chain construction to generate a sequence of single-use authentication tokens, once every single-use authentication token is verified by web server, it can not be reused again for subsequential communication, so it overcomes the limitation of nature of traditional session cookies that the authentication token is static and unchanged during its whole life circle. In this paper, the author assumed that in real world there exist two type of attackers, the passive attacker which can intercept all information like session cookies that communicate between web browser and web server, and can reuse the captured session cookie to impersonate victim to gain access to the session. The active attacker also can listen and intercept the communication information, in addition to this, active attacker can also fabricate message to web server, or stop message to prevent web server from receiving this message. It can also execute some application level attacks like cross-site scripting(XSS) attack or cross-site request forgery(CSRF) attack. By conducting experiments and analyze the security of OTC, the author showed that OTC has significant positive performance on preventing against the two type of attackers.

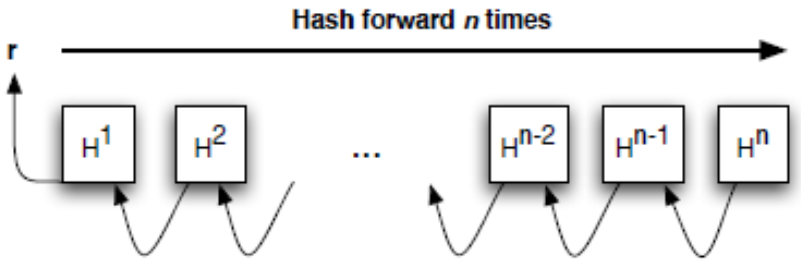

Figure 6: Hash chain

The OTC protocol balanced the vulnerability of session cookies and the expensive computations that come from deploying the HTTPS protocol over the whole website. It introduced a new type of session authentication tokens - single-use authentication token. These single-use authentication tokens generated by a proposed hash chains in Figure 6[DCA11], the hash chain generates a sequence of values by multiply using a cryptographic hash function H() to a random seed r, we have lots of candidates for the hash function like MD4, MD4, SHA-1. The security of the hash chain can be ensure by mathematically prove the nature of one-way property of cryptograhic hash function . Another highlight of design is OTC protocol used HMAC algorithm for additional security measures.

The whole work procedure of OTC protocol can be divided into two steps, that are setup and authentication. During the setup step, the web browser will lunch a initial secure connection with web server, and negotiate information with web server, based on these information, web browser will generate OTC credential and send it to web server for further authentication. For authentication step, web browser will use the OTC credential to generate a sequence of authentication tokens which work as session cookie for server side authentication. As these authentication tokens generated by a hash chain, so each will be different with others for every request, that is so-call one-time cookie. The authentication token along with other data will be appended to each request to be sent to web server, and a HMAC will also be appended together to be sent. By the verify the sent authentication tokens, web server can verify the authenticity of the request.

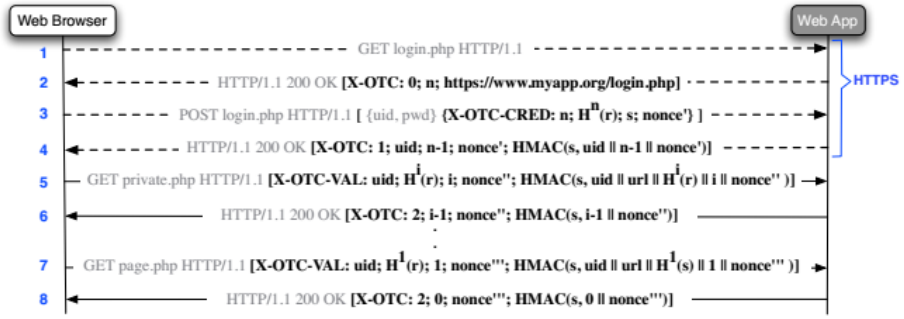

Figure 7: Workflow of OTC protocol

Figure 7 [DCA11]shows the whole workflow of OTC protocol. And Fig8 shows the formal symbol description of OTC protocol. In Fig7, from 1 to 4 are the setup step and 5 to 8 are the authentication step. In phrase 1, as OTC protocol relies on HTTPS protocol to establish the initial secure connection for generation of OTC credential, so web browser needs to send a request to web server to ask for login pages. As by now, the OTC protocol is not lunched, so HTTPS protocol is still needed to provide secure communication channel for transmit user's primary authentication credential, that are username and password.

In phrase 2. After web server received and verified the request, it will send back a 200OK response along with the requested login pages. The HTTP response contains a new OTC header called X-OTC, this header indicates that the web server support OTC protocol. The X-OTC contains three parameters, the first parameter 0 is a status flag which means the web server is ready to start the OTC protocol, and tell web browser to generate OTC credential. The second parameter n is the expected hash chain length that proposed by web server. And the third parameter is the url of requested login pages.

In phrase 3. Based on the information stored in response X-OTC header from web server, web browser generate OTC credential, and send the OTC credential to the login pages of web server along with user's primary authentication credential, that are username and password. The OTC credential is sent in the form of an new OTC header called X-OTC-CRED. The X-OTC-CRED also contains several parameters. The n is the hash chain length, the Hn(r) is the hash chain(the H() is hash function and r is the hash chain secret seed), the s is a shared session secret and a nonce. At

the same time, the web browser also store OTC state in browser side, and the OTC state contains the hash chain secret seed r, the shared session key s, the hash chain index i and the login url. The initially i = n-1.

In phrase 4. Once web server received the sent OTC credential and user's primary authentication credential(username and password), the web server first validate the user's primary authentication credential, if successful, web server will store OTC credential in memory in server side, and send back web browser a 200OK response along with a new X–OTC header to tell web browser the OTC credential is ready to use. The new X-OTC header contains a status flag 1, which means OTC credential is activated, a user id uid, a hash chain index i indicate the position of next expected hash chain value from web browser, a nonce and a HMAC, the HMAC is used to ensure the integrity of values in header, thus to ensure the the X-OTC header the web browser will receive is the complete and unmodified. The HMAC contains the session secret s, by this s, the HMAC can not be forged by attackers as they don't know the session secret s. After the complete the above 4 phrases, the setup step is finished and the OTC protocol is activated and lunched.

In phrase 5. When web browser want to ask for private information from web server, it will send a request to web server, the request include a new OTC header called X-OTC-VAL, the X-OTC-VA contains the user id uid, the hash chain value Hi(r), the hash chain index i, a nonce and a HMAC. When web server received the request, it will do the following operations. Firstly it will verify the sent hash chain index with the one in OTC credential that web server stored in server side, if they matched, then web server will verify the hash chain value by performing one time hash function on the hash chain value in the request, and match it with the one in its stored OTC credential. If successful, lastly, web server will verify the HMAC in the request by using session secret s to ensure the request is not intercepted and modified by attackers. If all measurements are successful, then web server can trust the request is valid and process the request. The web server will also update its stored OTC credential, for example update the hash chain value to the current received hash chain value. If any one of verification fails, the communication will be terminated and web browser will be redirected to the login pages to ask for authentication again.

In phrase 6. After processing requests, the web server sends back the private information and a 200OK flag to web browser. The response also includes a X-OTC header, that contains the status flag 2 that means successful OTC authentication, the hash chain index for the position of next expected hash chain value, a nonce

and a HMAC. After web browser received the response, it will also update its OTC state, for example update the hash chain index by deceasing one.

For phrase 7 and 8. The web browser can continue using OTC credential to generate one-time authentication tokens until the hash chain is exhausted, on other word i = 1.

Compared with HTTPS, the proposed OTC protocol has some improvements on several aspects that HTTPS can not achieve about preventing session cookie hijacking. As the OTC protocol use hash chain to generate a sequence of single-use authentication tokens, and these authentication tokens can be used as cookies, so for every communication, the authentication token is different, so even it is captured by attackers, if the authentication token has been verified by web server, it can not be reused any more, thus overcome the limitation that tradition cookie is static and unchanged during its life cycle. About the performance, according the result of experiments that the author conducted, it reveals that although the delay of OTC is longer than tradition cookies, however, the difference on performance is still can be accepted, so the affect of performance for OTC protocol can be ignore. And in order to deal with some advanced attacks like man-in-the-middle, as these active attacks can not only monitor and intercept OTC credential on the Internet, but also can forge or stopping OTC credential from reaching the web server. OTC protocol also use HMAC algorithm to ensure the integrate and confidential of OTC values, and attackers can not forge it as attackers do not know the session secret s, and because of the use of user id uid, attackers can not use captured OTC credential for other requests. Although these are some improvements for OTC protocol, it still has some limitations. As OTC protocol rely on HTTPS to initial the first secure connection to transmit OTC credential, it requires HTTPS to be configured correctly, and OTC protocol can not also prevent HTTPS attacks.

4.7 Sessonlock

Another approach that dealing with session cookie hijacking is proposed in [Adi08]. This approach makes use of URL fragment identifier to pass session authentication token between different web pages as URL fragment identifier is never passed in plain on the Internet, so attackers can not monitor and capture session authentication tokens, and thus can prevent against attacks like session cookie hijacking.

Sessionlock relies on HTTPS to lunch the initial secure session. After success veri-

fying user primary authentication credential(normally refer to username and password) and generate session identifier, sessionlock also generate a session secret based on the session identifier. The session secret is never transmitted over the network and is used by the web browser to generate authentication code for each subsequent HTTP request. The session secret is located inside url in a form of fragment identifier that appended the url, and can be passed from HTTPS login page to HTTP web pages, and also from one HTTP page to another HTTP page, as url fragment identifier need not to be transmitted through the Internet, so the session secret also does not need to be passed over the Internet, thus can prevent session cookie from being hijacked.

Sessionlock is relatively easy to implement with a small number of javascript programming and some server-side logical processing, and the technical components that it used are simple with approximately two components, that are fragment identifier and HMAC. In order to secure transmit session cookie, sessionlock choose fragment identifier as the medium to transmit session cookie as the nature of fragment identifier that never being passed through the network. And because the fragment identifier will be visually appended to url, so it can be implemented easily be scripting like javascript. And in order to meet the secure authentication and data integration, sessionlock adapt HMAC for web request authentication, and use the generated session secret as cryptographic key for HMAC operations. The whole work flow of sessionlock can be summarized into three steps that are session secret generation, session secret transmission and session authentication.

4.7.1 Session secret generation

The sessionlock needs to use HTTPS to establish initial secure communication and conduct user primary authentication to generate session secret. When a web browser want to setup a session with web server, it will first send a request to web server to ask for login pages, like *http://xxx.com/login*, after receiving the login page, web browser will send its user primary authentication credential(normally refer to username and password) to the login page for user authentication. After successful user primary authentication verification by web server, a session id will be generated, based on the session id, a session secret is also generated. Then web browser will send back the session secret to web browser by redirecting web browser to the page like.The session secret will be embedded in the url by the form of fragment identifier. The security of all above operations can be ensure as they all operate under HTTPS

by a secure communication channel, thus can protect against session secret being hijacked. By reaching this way, the session secret is generated and can be used for subsequent requests.

4.7.2 Session secret transmission

After got the session secret, when web browser want to send new request to web server or load new web pages, the session secret can be transmitted from HTTPS login page to HTTP pages, and from HTTP pages to HTTP pages, by appended to the url as the form of fragment identifier. The session secret can work as session authentication tokens for server side session authentication. Although by now there is no protection of HTTPS, the session secret can still be transmitted in a secure way as fragment identifier is never passed over the network, thus can prevent against session cookie hijacking. There are two ways to implement session secret transmission. If implemented by native javascript code, the session secret needs to be appended to url as a fragment identifier for every HTTP request, this is because each HTTP request will update the whole web page and the url will be changed. However if it is implemented in a ajax way, as ajax request is executed in a synchronous mode and in the background, a ajax request will only update a partial of the web page instead of al,l and the url will keep the same, so it is not necessary to repeat append session secret to url for every HTTP request, only need to append to url for the first page initial. Actually, sessionlock is more easy and suitable to be used with AJAX applications.

4.7.3 Session authentication

With the session secret, each HTTP request must be augmented by adopting HMAC algorithm. When web browser sends HTTP request to web browser, it will append a timestamp as the first parameter to the end of request(the request also include the session secret appended to the end), and HMAC the entire request and append the value of HMAC to the request as the second parameter, after all these done, the current request is the final request to be sent to web server. For hash operation by HMAC, the session secret is used as the cryptographic key. Once the web server received the request, it will also use the session secret to authentication the request. Interestingly, if it is a ajax request, the sessionlock will work more transparent. By using ajax we can achieve the same goal, however, these additional parameters will

not be shown in url for ajax requests, and will give user better experience.

As because sessionlock use url fragment identifier as vehicle to transmit session secret, and url fragment identifier is explicitly appended to the url, so it may entirely exists a possibility of lost of url fragment identifier. For example, if a user type the url manually, it is highly possible that he doesn't type the url fragment identifier, thus make sessionlock can not work correctly. The author considered this possibility and proposed a way to solve it. Sessionlock use an HTML IFRAME tag to solve this problem. When a web page notices the missing of url fragment identifier on a url, it will use a invisible IFRAME to recover the url fragment identifier, in other words, the session secret, to the current url, and then redirect the web browser to the HTTP page with having url fragment identifier inside it.

The author also proposed a way to implement sessionlock without using HTTPS protocol. As the previous introduced sesisonlock use HTTPS to for initial communication channel setup, and by the secure communication channel to transmit session secret for later session authentication, so the author also proposed a way by using Diffie-Hellman key exchange[DH06] algorithm to generate the session secret, thus can make sessionlock work without the help of HTTPS. So the workflow of new sessionlock is like the following:

1. Firstly, the web server assign a session cookie for web browser.
2. Then, the web browser use Diffie-Hellmain key exchange algorithm to generate the session secret between web browser and web server.
3. The generated session secret is used as the cryptographic key for HMAC algorithm that used by sessionlock. And the session secret is transmitted from one HTTP page to another in the form of url fragment identifier.
4. And if the session secret is lost, sessionlock can simply perform the Diffie-Hellman key exchange algorithm to generate a new session secret.

The sessionlock is a lightweight implementation, it only requires a small number of javacript code, and some server side logical programming. The principle of sessionlock is simple and easy to understand. Apart from easy implementation, as it only require javascript, it almost supports all major web browsers. However, it also suffers from two important limitations. One of which is the requirement of javascript. The javascript requirement is both the strongpoint and the limitation. As it depends on the deployment of javascript, it can not work by web browsers which do

not support javascript. And also as the requirement of using javascript, it brings in new flaws, that is sessionlock can suffers from attacks which make use of javascript. The second limitation is it can not defend against active attacks. As active attackers can inject malicious code into url, and intercept session secret, modify it and replay to impersonate victim to access the session with web server, the sessionlock can not prevent these kind of attacks, however, it is enough to defend against session hijacking.

4.7.4 Rolling code

In paper [CB11], the author by referring the rolling code technology that is used to prevent intercepting, capturing and replaying a code by attackers to open a garage door, to make use of this technology in the field of preventing session cookie hijacking. The method that proposed is called rolling code method.

In this paper, the author discussed three main security purposes for cookie, that are authentication, integrity and confidentiality. In order to achieve the three security purposes, the author proposed three security levels that each of which meets some requirements of these three security purposes at some extent, and what security level is actually used in practice is up to the choice of website administrator, thus web site administrator can balance the tradeoff between level of security of performance of website.

The proposed rolling code method relies on the HTTPS protocol to lunch the initial secure communication for user primary authentication, and also for transmitting some necessary secret values that be used in rolling code method. after finishing user primary authentication and the initial rolling code parameters exchange, the web server will continue work without the security protection of HTTPS, and begin from that point, rolling code method is starting the protect the security of session cookie from attacks. So when web browser want to setup a session with web server, it will first establish an HTTPS connection, after verification of user primary authentication(the username and password), web server will use some algorithms to generate two secret values, we call them seed and d, they both take 160 bit long and very important as they associate the whole process of rolling code method. The algorithm of generating these two secret values is up to the web server' s choice, after generation, the two secret values will be sent to web browser. The key point of rolling code method is the rolling code, which is located in cookie in the form of parameter, and is named as RollingCode.key. the RollingCode.key is transmit-

ted between web browser and web server for session authentication. Rolling code method use three steps to generate RollingCode.key, when get the seed and d secret values, firstly, it performs a certain hash algorithm on d, and assign the hash value to d, if presented by formula, it will be like d = hash(d). After doing this, we we get a completely new value d, and then XOR this new value d with the seed, and store the result in a temporary variant tmp, this step can be presented by formula like tmp = XOR(d, seed), this a one way calculation as even if adversaries intercept the tmp, it is almost impossible to break it into d and seed, thus can not hack this method. Lastly, we hash the generated tmp value to get a 160 bit new value, and assign the new value to RollingCode.key. Finally, we complete the RollingCode.key generation and we can use it for authentication. Later if web browser continue to communicate with web server, it will send the request along with the cookie to web server, on web server, it will perform the same operations by using its own seed and d to generate the key, and then verify the generated key with the key sent from web browser to see if they can be matched, if matched, web server can make sure the identify of the request come from the real sender and process the request and response to web browser.

The hash algorithm used in rolling code method can be assigned to a specific one by web server in advance. However, rolling code method allows to assign a default hash algorithm, if web server does not assign a specific one, then both parties can use the default hash algorithm, the common hash algorithms can be used such as SHA-1 or AES-128.

The key component of rolling code method is RollingCode.key, which can meet the requirement of authentication. In order to meet other advanced security requirement like integrity and confidentiality, the author supplement the rolling code method by proposed three security level.

- Level 1 is the lowest security level, which is aimed to meet the authentication requirement. As described before, the basic function of rolling code method, this is the RollingCode.key, can enough to meet the authentication requirement as adversaries can not break RollingCode.key and do the reverse hash operations. RollingCode.key = hash(XOR(seed, hash(d)))
- Level 2 is the medium security level and ask for authentication and integrity requirements. In order to meet the integrity purpose, rolling code method hash the entire cookie, including the RollingCode.key, to prevent the malicious modification by adversaries. and then XOR the hash value with d, and hash the

result again, by doing this operation to protect the hash value to be recalculated by adversaries. Finally assign the result in cookie as a parameter which is called RollingCode.integrity. by formula, it looks like RollingCode.integrity = hash(XOR(hash(entire cookie), d)).

- Level3 has the highest security requirement, based on the level2, level3 also include the confidentiality requirement. Under this level3, rolling code method encrypt the whole cookie by using AES algorithm with a private key, the private key is generated by XOR the seed and d secret values, by formula like key = XOR(seed, d). The value of encryption is stored in the cookie and called RollingCode.payload.
- And if the session secret is lost, sessionlock can simply perform the Diffie-Hellman key exchange algorithm to generate a new session secret.RollingCode.payload = encrypt(entire cookie, key)

When web server receive requests from web browser, it will perform some actions on these generated values on cookie to verity security standards according to different security level that web administrators set. For authentication and integrity, web server will perform the same actions described in level 1 and level 2 by using its own seed and d secret values, and to generate RollingCode.key and RollingCode.integrity to match with received RollingCode.key and RollingCode.integrity, thus to ensure the authentication and integrity. For confidentiality, the web server will decrypt the encrypted content to get the payload.

4.8 VPN

Another common way usually adopted by companies or organizations to establish secure network communication channel and prevent private network traffic being hijacked is VPN. Figure 8 shows the overview of VPN. VPN is shorted for virtual private network. According to the name, we can know that VPN is a group of network hosts that being gathered to form a network, and the network is being encrypted. However, different from traditional private network, VPN is a virtual private network, that means there is not a physical specific private communication channel for VPN, actually, VPN is deployed across public network such as Internet. VPN enables users to send and receive data across a public network as if it were connected to a private network, and thus benefit from the same security level of private network. Another different from traditional private network is that VPN does

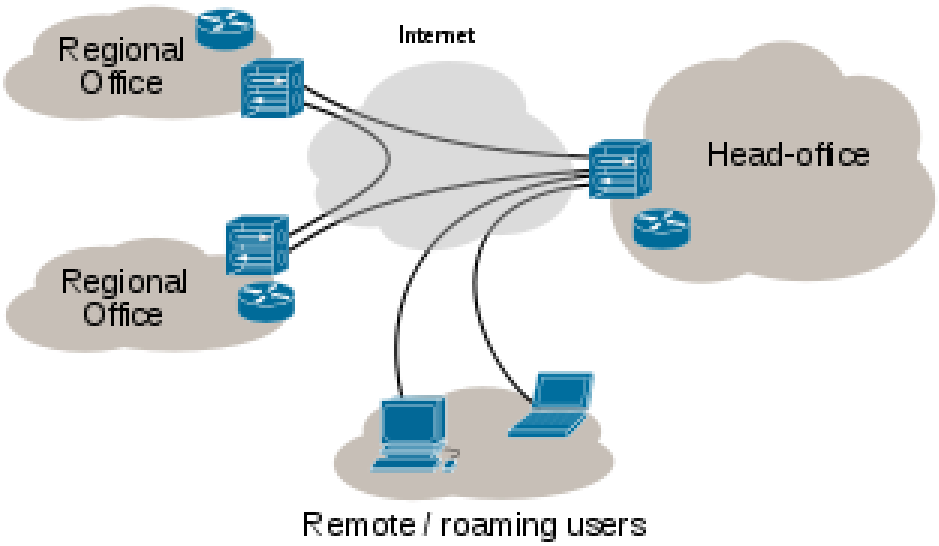

Figure 8: VPN

not require these machines in the VPN to gather together geographically, actually, in most of time the hosts in VPN are discrete, each host is far away from others. VPN allows businesses to connect to a remote datacenter in a safe way by across its encrypted communication channel, or users by using VPN to get access to a remote network resource securely, although they all use an untrusted public network.

A VPN is created by using some dedicated secure connection protocols such as PPTP (Point-to-point Tunneling Protocol) or Internet Protocol Security (IPSec), thus to establish a virtual point-to-point connection. when you plan to setup a VPN to a remote web server, you usually can use a VPN client on your computer, sometimes the VPN client is in the form of a linke on a website. Then by log in with your credentials, usually refer to a username or a password, then your computer will exchange trusted keys with the remote web server. once the authentication is successful, then you has established a VPN with the remote web server across the Internet, which means the communication channel between you and the remote

web server is encrypted and secured from network attacks such as session cookie hijacking.

4.8.1 Pros and cons

Since the introduction of VPN, VPN is especially suitable and useful in business area, it allows business partners to communicate with each other from a remote location through a secure dedicated network which follow some secure protocols, and compare with traditional private network that needs to deploy dedicated physical hardware, VPN is much cheaper and can decrease the cost for companies. However, VPN is not suitable for every situations, for example it is not suitable for mobile workers, and not suitable for wireless connection cause these may cause some security issues for the network. Beside this, VPN also has some other down sides. The below discusses the pros and cons of VPN from some aspects.

Pros: One feature of VPN which different from traditional private network is that it does not require hosts in VPN to gather together in a geographic way, the structure of VPN is diverse and even in a discrete way, for example, people who is at home can be easily to connect to a remote resource without sacrificing security, this feature make VPN can be used in many scenarios especially for business, people can work from home and access to company's dedicated private network as if he is in company, thus enable for remote work.

Cons: However, VPN is not always suitable for any scenario. For example, VPN is not suitable for mobile workers as by using mobile devices to establish connect to VPN may cause some security issues. VPN is also not suitable for wireless connections as also because of security concerns. In order to make it suitable for mobile worker, some extra effort or additional solutions are needed to integrated with VPN to solve the security issues.

The cost of deploying a VPN for secure communication is much cheaper than setup a traditional private network, this is main because VPN is utilize the public network plus some dedicated secure protocols to virtualize a private network, and does not need to setup some dedicated hardware to establish a private network, and can also save resource. By using VPN can significantly decrease the cost for many companies.

Compared with other remote communication methods, VPN not also provides a remote communication service, but also offers secure communication services by utilizing some secure secure protocols. This additional feature is very important

and especially for business companies, as the data of business companies is security-sensitive, and needs to be protected from leakage.However, the high security bonus also bring more extra effort for implementation and configuration of VPN, and make the network become more complex, thus require more professional knowledge for implementing and configuration for best use of VPN.

5 Random Cookie protocol

In this section, we first work on a session cookie hijacking demonstration, and then we present why we design random cookie protocol(RCP), what aspects we want to achieve by design random cookie, how we design this protocol, including its design, implementation, experiments and the security analysis. Firstly, we discuss the reasons of designing the random cookie protocol and the different security levels we want to achieve according to the designed protocol, mainly talk about two different security levels that are authentication level and integration level, and also discuss why it is hard to achieve confidentiality level. Then we give a design of random cookie protocol. Next, based on the design, we give a detail description of how each steps of random cookie protocol perform. We also analyze why the random cookie protocol meets its security requirements, including the authentication and integration requirements. Finally, we conduct some experiments and analyze the performance of random cookie protocol based on the experiment results.

5.1 The demonstration of session cookie hijacking

The cookie hijacking has been a very known and frequent used network security vulnerability since the introduction of HTTP cookie in 1990s. Although the principle of session cookie hijacking is relative simple, the prevent for it has gain little improvement. Now, a lot of tools have been developed such as "Ferret and Hamster" and "Firesheep", these tools can automatic perform session cookie hijacking by monitoring the network traffic, capture data and modify it to replay to access victims' session. Actually, session cookie hijacking can also be performed manually. In the below, we will demonstrate the process of session cookie hijacking by manually performing it.

For performing session cookie hijacking, the first step is to monitor and capture HTTP traffic. Actually, the HTTP traffic monitor and capture is an easy process

in the previous when the network hosts are still relied on hub for transmission. Under that situation, as all hosts in that network will all connect to the hub, all network traffic will pass through the hub, when hub receives a packet of data from on connected host, it will broadcasts the data packet to all other connected hosts without the real final destination, because of this nature, the attackers need only to plug in to the network, monitor and capture traffic data from any host. Later when the switch is designed and start to be used, the situation is different. Like hub, the switch will connect all hosts in the network to each other, however, the difference from hub is when switch receives a packet of data, it can determine what host the data packet is intended to and will send this data packet to the desired host, it will not broadcast traffic to all connected hosts like hub, so when the switch appears, the monitor and capture HTTP traffic becomes more complicated. However, there are still some tricks can be made use of by attackers to intrude to the network such as main-in-the-middle attack. When the WiFi is proposed and become popular, as the nature of penetration, it offers us more ways to hack into the network.

For this session cookie hijacking demonstration, we used a web tools called web developer. The web developer tool is a plugin that support some main web browsers like firefox or chrome, by using this web plugin we can easily to obtain more web information such as cookie. For website, we use a chinese website called douban.com for demonstration. We first login the website by using firefox web browser, The Figure 9 show the douban.com website after login.

Figure 9: Website login

The second step for session cookie hijacking is to read, add or modify the captured session cookie. By using the web developer tool, we can capture and see the session cookie for the victim's session. The Figure 10 shows the session cookie. From the fig11 we can see that session cookie name is dbcl1, and the session cookie value is 128531458:EwnT0dbvMtg. By now, we successfully got the victim's session cookie. Then we can add or modify it to impersonate the victim to access the victim's session.

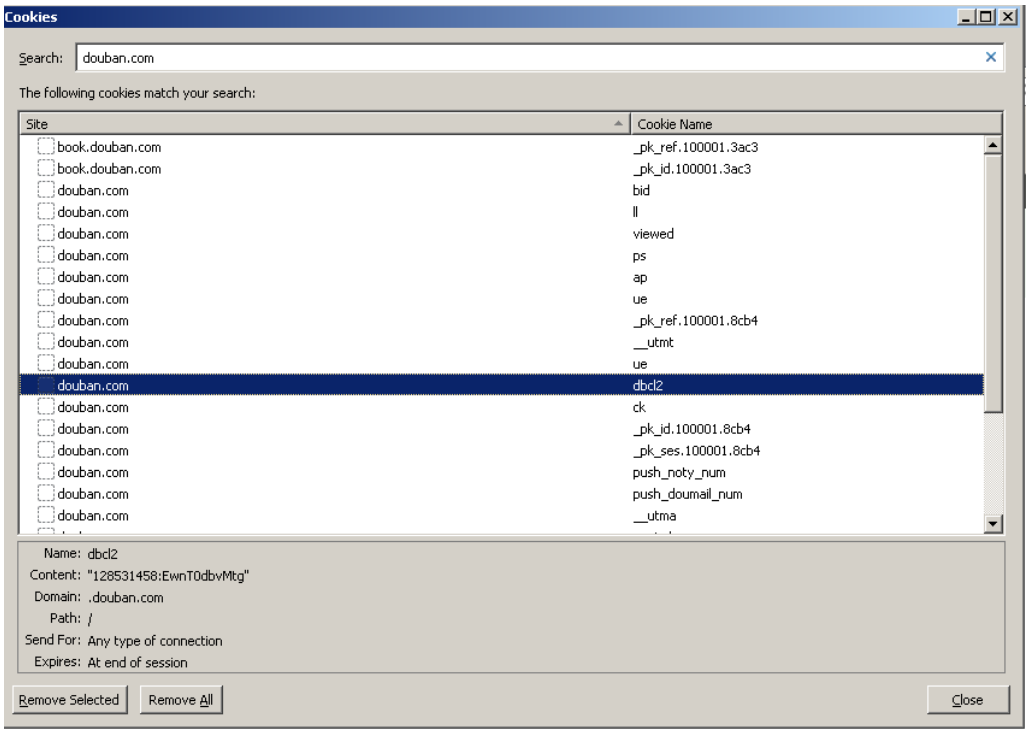

Figure 10: Session cookie

Next we open another web browser chrome, and open the douban.com website without login. For this chrome website, as we don't login the website by using this browser before, so the chrome doesn't contain the session cookie for the website. Then, by using the web developer, we add the captured session cookie to chrome. The Figure 11 shows the addition of captured session cookie.

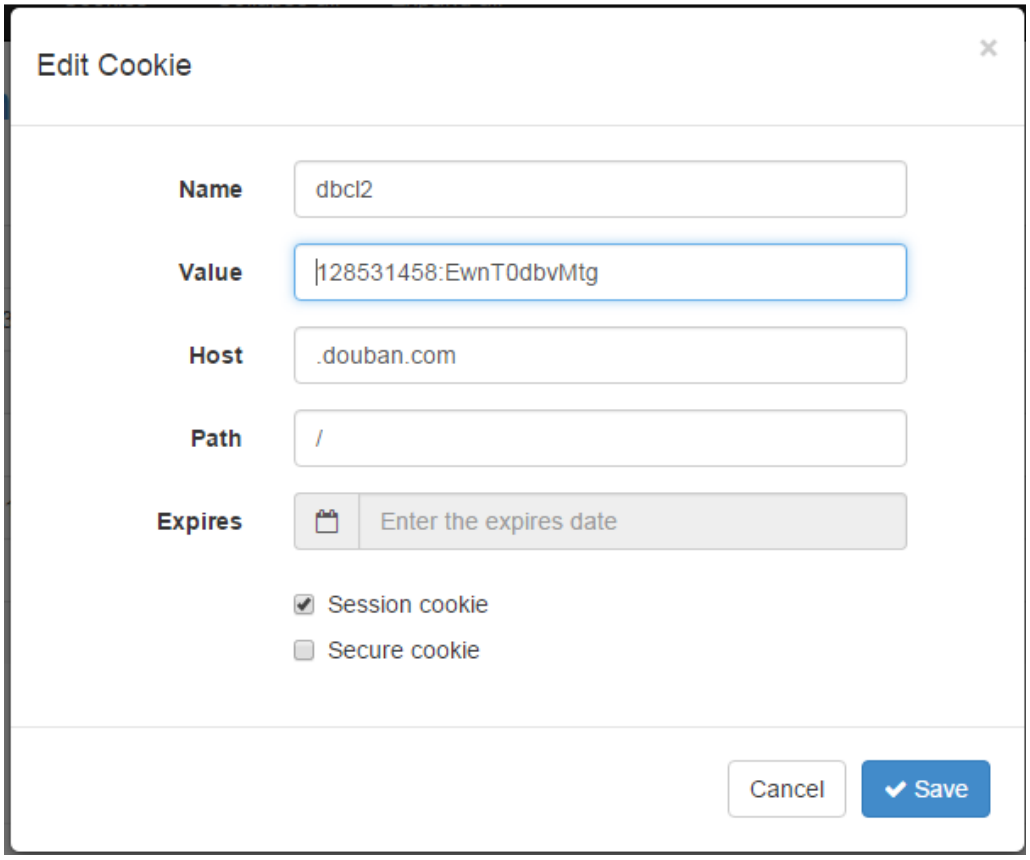

Figure 11: Captured session cookie

5.2 Desired goals

In real world practice, one kind of way to measure the performance of a network protocol is to see whether it can meet some critical goals, for security aspect, it mainly includes authentication, integration, and confidentially aspects. For authentication requirement, if a network communication protocol can meet authentication, it means the server can trust the request is sent from the real client instead of some third parties or potential adversaries. For integration, if a network communication protocol meet this, which means the server can ensure the sent request is the original request and is not being modified by others. When it comes to confidentiality, the protocol which meets the confidentiality can prevent the communication traffic from leakage, thus to ensure the security of communication traffic itself. So for the random cookie protocol, we also try to make it meet all of the three main requirements, however, the confidentiality is relatively hard to achieve because this aspect

may take too much cost and will seriously affect the protocol performance

- Authentication

  Authentication is the basic requirement for a network communication protocol, and is also a important part for our random cookie protocol. Because of the awareness of computational expensive for HTTPS, most of websites try to avoid using HTTPS over the whole websites by just deploying it for login pages, thus to make the negative affect for performance to be minimal. In addition to the performance concern, the traditional session cookie used for authentication is transmitted over network in a plain way, and it is unchanged during its life cycle until reach the expiration time, make the session cookie unsafe and have the risk of being stolen, so our random cookie protocol not only meet the authentication that the session token can prove the identify of client, but also can ensure that even if the session token is stolen, the adversaries still can not use it to replay and impersonate the victim to access the session.

- Integration

  Integration means the communication traffic is original and is not being modified by others. It can be achieve by using hash algorithms, normally we hash the entire or certain part of the request, and append the hash value to the end of the request and send it to the server side, during server side, it perform the same hash operations based on the request(exclude the appended hash value), and match the new hash value with the sent one to verify the integration, this way is made use of the one way nature of hash algorithm. For the random cookie protocol, we make use of HMAC algorithm to achieve the integration.

- Confidentiality

  Meet confidentiality for communication means the communication traffic can be in a safe way and in confidential, and can be protected from being read by other unauthorized parties. However, this part is relative hard to meet. As we know the most straight way to meet confidentiality is to encrypt the sent request, but this will cause expensive computational cost and thus can affect the performance. The HTTPS can meet confidentiality, however the reason why most of website try to avoid using it over the whole network is just because of the performance awareness. So for our random cookie protocol, as our main attention is focused on how to prevent against session cookie hijacking, so we

leave the confidentiality requirement alone, and just meet the authentication and integration.

5.3 Design

Based on one-time cookies[DCA11],we innovate the author's idea by changing one-time cookies to completely random cookies. The reason why our protocol is called random cookie protocol is because for our protocol, the session cookie used for every request is different, and this is also the main point and main contribution for our solution. For random cookie, even if one adversary can capture the session cookie, he can not use it any more as the session cookie of out of date. Our random cookie protocol consists of two phrases that are negotiation and authentication. During the negotiation phrase, client side and server side will negotiate and exchange some necessary values that used for later authentication, such as seed and r for later hash algorithm, and these values are transmitted by being stored in protocol header. For authentication phrase, it is mainly for using the generated random cookie for each communication authentication. Our design will focus on and try to meet the authentication and integration requirements for the random cookie protocol.

- The design for authentication

  Random cookie protocol needs the help of HTTPS for negotiation phrase. During the negotiation phrase, after finish user primary authentication, the web client and web server will negotiate and exchange two values that we can them seed and r. The whole negotiation phrase is protected by HTTPS protocol, so the safety of values seed and r can be guaranteed and we can trust them for later authentication. For authentication phrase, for each request, web browser make use of seed and r and based on hash algorithm to generate a hash value and use the value as session cookie, and send it to web server for authentication, each time after successful authentication, the web server will also generate a new r and send back to web client, so for the next request, the generated hash value is random and different, and the session cookie is different, so even if the session cookie transmitted on the network is captured by adversaries, it can not be used again as the next expected session cookie is different.

- The design for integration

Our random cookie protocol use HMAC to implement integration. HMAC is short for hash-based message authentication code, it is usually be used as a message authentication mechanism to verity the integration of the message being sent. HMAC is based on hash algorithm, and by using a share key and the message to be sent as input to generate a hash value used as a message digest, and append the message digest to the message and send them all to the server side for integration verification. For our solution, when the session cookie is generated in web browser side, random cookie protocol HMAC the whole request that include the message to be sent and the session cookie by a share key, the share key is generated by XOR the seed and r, and append to generated message digest to the end of the request and send the request to web server. When web server receive the request, it will do the same operation that HMAC the request(exclude the appended message digest) to match the message digest sent from web browser to verify the integration.

5.4 Formal description

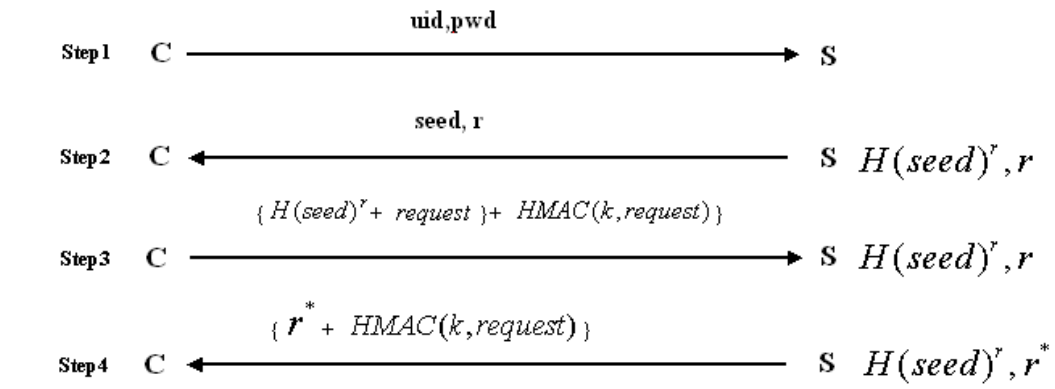

Figure 12: Formal description of random cookie protocol

Figure 12 shows the formal description of random cookie protocol and the Figure 13 presents the parameter definition. In Figure 12, the step1 and step2 represent the negotiation phrase, and step3 and step4 represent the authentication phrase. The Figure 12 assumes that the random cookie protocol is never used before, so when

uid. pwd: user primary credential

seed: the original value to be hashed by hash algorithm

r: the number of hash

k: the share key for HMAC, the k = XOR(seed, r)

Figure 13: The parameter definition

its first time use, it first setup the credential(the seed and r) by negotiation between web browser and web server, then based on the credential to instead the traditional authentication cookie for session authentication.

For step1 of negotiation phrase, when the web browser establish a connection with web server and ask for the login page, and the web server response to web browser by returning the login page, the negotiation phrase starts. The web browser sends its userid and password over HTTPS to web server for user primary authentication, and the HTTPS is also required here to provide secure connection for server side authentication.

For step2, when the web server receive the user primary credential(the userid and password), it do the user primary authentication, if it's successful, the web server will generate two values by some certain random number generation algorithms. The first value is called seed, as its name suggests, it is the original values which will be hashed by hash algorithm for several rounds to generate the random session cookie, the second value is call r, it indicates the number of rounds of hash operations. After generating the seed and r, the web server will hash the seed for r times, and stores the H(seed)*r and value r in memory in server side, and then response to web browser by sending back the generated seed and r to web browser over HTTPS, when web browser received the seed and r, the negotiation phrase is finished, and now, both web browser and web server has the random cookie credential(the seed and r) that used for up coming authentication.

For step3 of authentication phrase. Once the negotiation phrase is finished, for up coming communication, the random cookie protocol is replaced for traditional session cookie to do the session authentication, and HTTPS protocol is not needed any more as random cookie protocol can prevent web session from being stolen and replayed by network adversaries. When web browser send request to web server, it will first use the received seed and r by hashing the seed by the predefined hash algorithm for r times like H(seed)*r, use the hash value as the random cookie, and send the random cookie along with the request to web server. In order to ensure the integration, HMAC algorithm is used here to provide integration service. The HMAC hash the entire request plus the generated random cookie to generate a message digest by a share key, and use the message digest to verify the integration. The share key used by HMAC is generated by XOR the seed and r as k = XOR(seed, r), so the HMAC is like HMAC(k, request + H(seed)*r). Then append the generated message digest to the end of request and send them all to the web server.

For step4, when web server receive the request sent from web browser, it will first verify the Integration, then authentication. For verify integration, it will use the seed and r that stored in server side to XOR(seed, r) to generate the share key, and by using the share key to HMAC the request(exclude the message digest) to generate a new message digest, and compare it with the message digest which generated in web browser side, if it match means the request is not modified, thus can verify the integration. For authentication, web server will compare the random cookie sent from web browser with the one stored in web server, if it matches, then means the sender is the real sender and web server can trust the received request, if don't, web server will redirect web browser to the login page and ask for conduct user primary authentication again. When finished authentication, web server will use the random number generation algorithm to generate a new r, and calculate new H(seed)*r and update the old one, then response to web browser by sending back the new r and a new HMAC to ensure the integration of r and response like HMAC(r, response). The whole authentication phase is finished, if web browser send request again, it will repeat step3 and step4 for communication by using random cookie.

5.5 Security analysis

In this part, we will discuss and analyze the security properties of random cookie protocol, and how it prevent against session cookie hijacking.For random cookie protocol, every time a new r is generated by random number generation algorithm,

by the random r, a new and unique random cookie is generated, therefore, for every request, a one-time use session cookie is sent by web browser and verified by web server, and it can not be reused again as the session cookie for next time authentication is different. As the result, some passive attacks such as session cookie hijacking can not play any more.

Some active attacks may do more advanced attacks, such as man-in-the-middle attack, it not only can steal the session cookie, but also can prevent it from reaching to the web server, so random cookie protocol use HMAC to solve this problem. Random cookie protocol hmac the entire request and the generated random cookie, and tie the message digest to the request to send to web server, even if an adversary can capture the request, it can not forge the message digest as he does not know the share key, and the share key is also changed for every request by generated based on the one-time user.

5.6 Experiments and results

In order to analyze the performance of our random cookie protocol, we implemented the random cookie protocol and wrote a benchmark program to test the performance. The program is implemented by c programming language. In order to simplify the experiment, we didn't consider using HTTPS protocol for initial user primary authentication, and assume the initial user primary authentication is successful finished. Our attention is focused on the performance of how random cookie protocol generate credential, transmit, and do the authentication after the initial user primary authentication is finished. To contrast with HTTP protocol, we also implemented a simple program to compare the performance of random cookie protocol with traditional HTTP protocol.

For each protocol, we loop 10, 100, 1000 and 10000 iterations and recorded the average time in milliseconds per iteration, and each iteration is refer to one communication that from web browser to web server. For each number of iteration, since now the way to access network is diversity, we chose two most common platforms that on a Android mobile phone and a desktop PC, and we tested both on them.

The Figure 14 shows the average time of 1000 rounds of loop for random cookie protocol and HTTP protocol on a Android mobile phone platform and on a desktop PC. From the Figure 15 we can see that the performance of HTTP protocol is slightly better than random cookie protocol.

Platform	Android mobile(ms)	Desktop PC(ms)
HTTP	0.168	0.021
Random cookie protocol	0.295	0.064

Figure 14: The average time of 1000 rounds of loop for random cookie protocol and HTTP protocol

Android	10 round	100 round	1000 round	10000 round
HTTP	0.169	0.170	0.168	0.153
RCP	0.298	0.301	0.295	0.254

Desktop PC	10 round	100 round	1000 round	10000 round
HTTP	0.041	0.029	0.021	0.025
RCP	0.114	0.077	0.064	0.072

Figure 15: The average time of HTTP and random cookie protocol on Android and Desktop PC

The Figure 15 shows the average time of HTTP and random cookie protocol for 10 round, 100 round, 1000 round and 10000 round of iterations on Android platform and desktop PC respectively. From the data we can see that the time for random cookie protocol is almost 2 times than the time of HTTP on android platform, however, as the unit is millisecond, the performance of random cookie protocol is still relatively good. Similar to the comparison for android platform, for desktop PC, the running time of HTTP protocol is still less than that of random cookie protocol, however, the difference is relative slight and can be acceptable, comparing with risk of session cookie hijacking for HTTP, the security guarantee of random cookie protocol is enough to offset the slightly disadvantage on performance.

6 Conclusion

In this thesis, we first talk about network security, including wired network security and wireless security, we discuss the current threads and attacks we face for the

wired network and wireless network, then we introduced the current solution for these threads and attacks. Among these threads we mainly focus on the session cookie hijacking thread, then we discussed some background knowledge for this attack including what is cookie, what is session, the principle of session cookie authentication and the threads for session cookie, we then talked about the principle of session cookie hijacking, and proposed some current solutions for this attack, such as HTTPS protocol, one-time cookie, sessionlock and rolling code, and analyze their pros and cons. Next, we demonstrate the whole process of session cookie hijacking by manually perform it, lastly, we proposed our solution which called random cookie protocol, and we did experiments to compare the performance with HTTP, although the result showed the performance of random cookie protocol is less than HTTP, however, the difference is slightly and can be acceptable, and we think the guarantee of session cookie security is worth enough to offset the additional 100ms performance.

References

Adi08 Adida, B., Sessionlock: Securing web sessions against eavesdropping. *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, New York, NY, USA, 2008, ACM, pages 517–524, URL `http://doi.acm.org/10.1145/1367497.1367568`.

BHH+10 Bittau, A., Hamburg, M., Handley, M., Mazières, D. and Boneh, D., The case for ubiquitous transport-level encryption. *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, Berkeley, CA, USA, 2010, USENIX Association, pages 26–26, URL `http://dl.acm.org/citation.cfm?id=1929820.1929855`.

CB11 Cashion, J. and Bassiouni, M., Robust and low-cost solution for preventing sidejacking attacks in wireless networks using a rolling code. *Proceedings of the 7th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Q2SWinet '11, New York, NY, USA, 2011, ACM, pages 21–26, URL `http://doi.acm.org/10.1145/2069105.2069110`.

CDW06 Coarfa, C., Druschel, P. and Wallach, D. S., Performance analysis of tls web servers. *ACM Trans. Comput. Syst.*, 24,1(2006), pages 39–69. URL `http://doi.acm.org/10.1145/1124153.1124155`.

CMWZ09 Chen, S., Mao, Z., Wang, Y.-M. and Zhang, M., Pretty-bad-proxy: An overlooked adversary in browsers' https deployments. *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, 2009, URL http://dx.doi.org/10.1109/SP.2009.12.

CRW07 Chen, S., Ross, D. and Wang, Y.-M., An analysis of browser domain-isolation bugs and a light-weight transparent defense mechanism. *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, New York, NY, USA, 2007, ACM, pages 2–11, URL http://doi.acm.org/10.1145/1315245.1315248.

DCA11 Dacosta, I., Chakradeo, S., Ahamad, M. and Traynor, P., One-time cookies: Preventing session hijacking attacks with disposable credentials, 2011. http://hdl.handle.net/1853/37000. [12.01.2011]

DH06 Diffie, W. and Hellman, M., New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22,6(2006), pages 644–654. URL http://dx.doi.org/10.1109/TIT.1976.1055638.

JB08 Jackson, C. and Barth, A., Forcehttps: Protecting high-security web sites from network attacks. *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, New York, NY, USA, 2008, ACM, pages 525–534, URL http://doi.acm.org/10.1145/1367497.1367569.

HJB10 J.Hodges, C.Jackson and A.Barth, Http strict transport security (hsts), 2010. http://tools.ietf.org/html/draft-hodges-strict-transport-sec-02. [12.01.2011]

Gun10 Lackner, G., A comparison of security in wireless network standards with a focus on bluetooth, wifi and wimax, 2010. http://www.herongyang.com/PKI/HTTPS-Server-Authentication-Process.html. [12.17.2010]

SDOF07 Schechter, S. E., Dhamija, R., Ozment, A. and Fischer, I., The emperor's new security indicators. *Security and Privacy, 2007. SP '07. IEEE Symposium on*, SP '07, Washington, DC, USA, May 2007, IEEE, pages 51–65, URL http://dx.doi.org/10.1109/sp.2007.35.

DTM08 Turner, D., Mack, T. and Low, M. K., WWW-symantec internet security threat report, 2008. http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_exec_summary_internet_security_threat_report_xiii_04-2008.en-us.pdf. [07.12.2008]

Her16 Yang, D. H., Https server authentication process, 2015. http://www.herongyang.com/PKI/HTTPS-Server-Authentication-Process.html. [12.01.2011]